Applied!

Data & Network Security

Behnam Amiri

ans.dailysec.ir

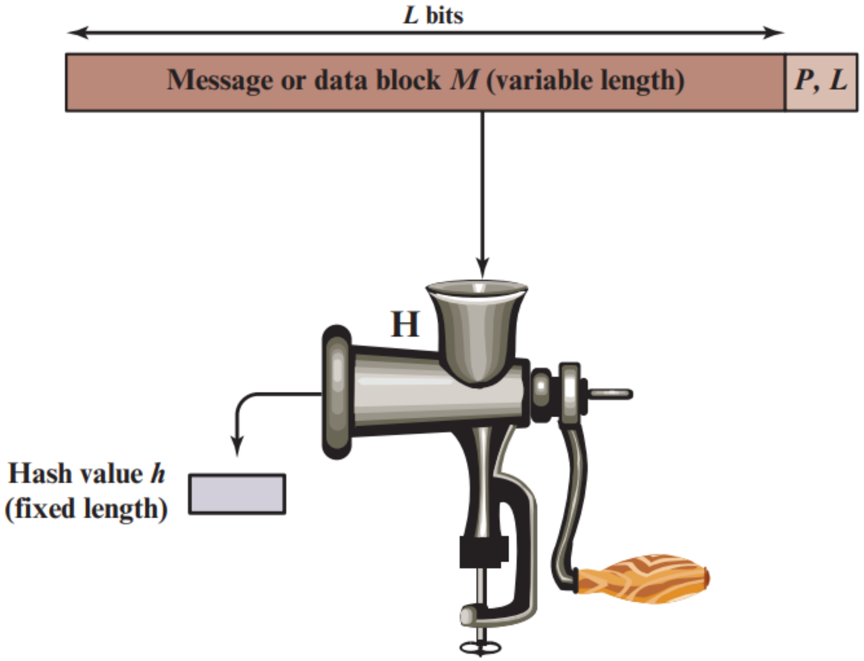
aNetSec.github.io

Spring 2025

Hash Functions

Hash Function

- A hash function H accepts a variable-length block of data M as input and produces a fixed-size result h = H(M)

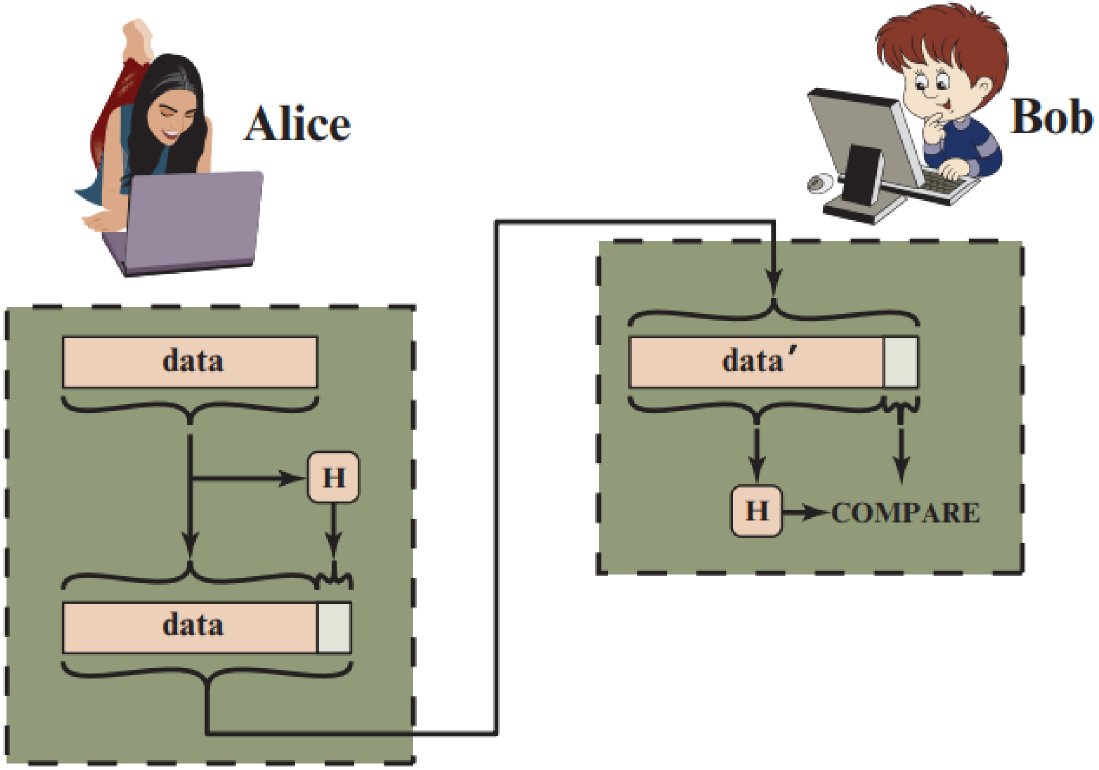


P, *L* = padding plus length field

Figure 11.1 Cryptographic Hash Function; $h = \mathrm{H}(M)$

Applications Of Hash Functions

- Check integrity of message
- Check message not changed in transfer



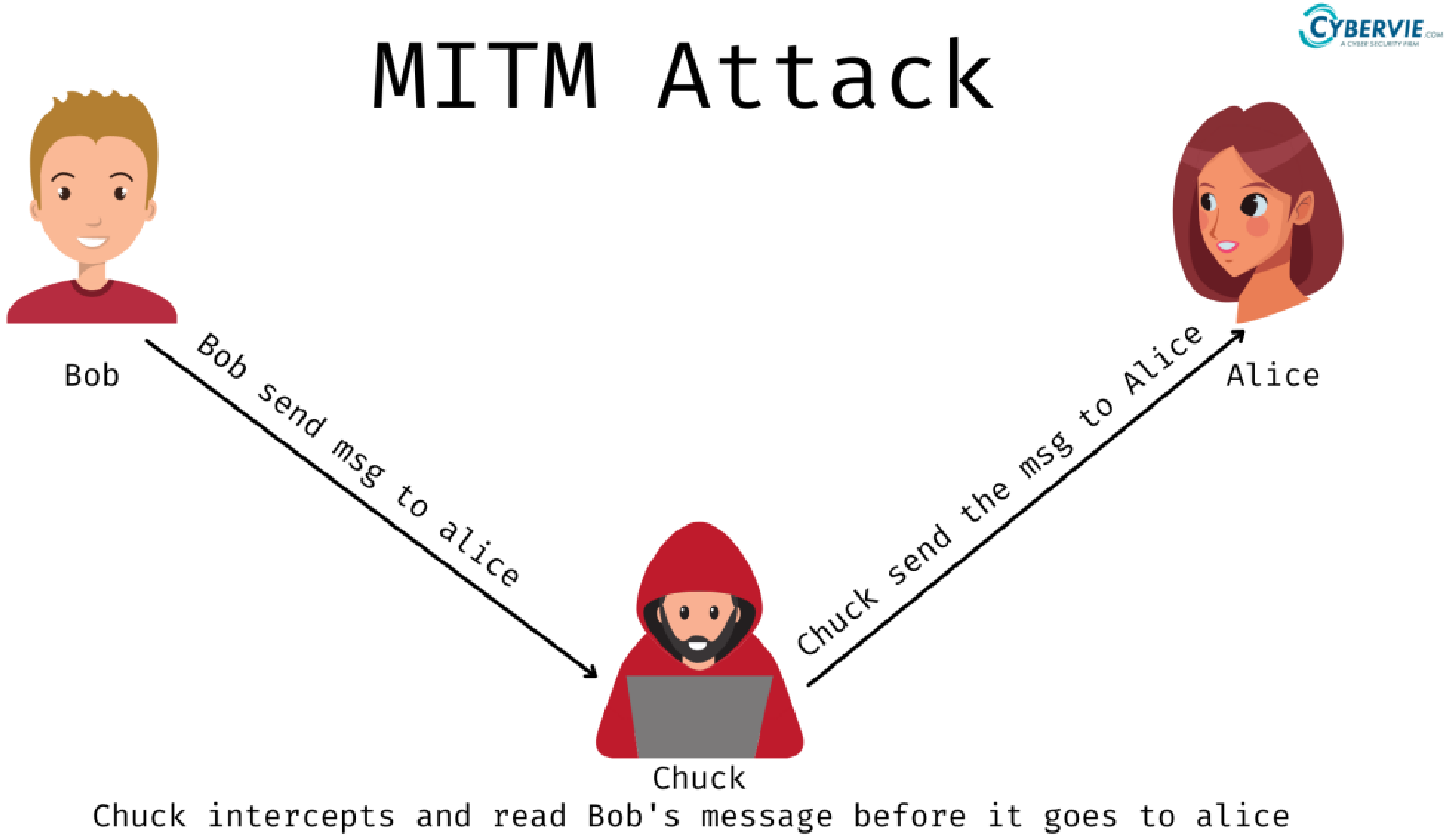
(a) Use of hash function to check data integrity

Sniff

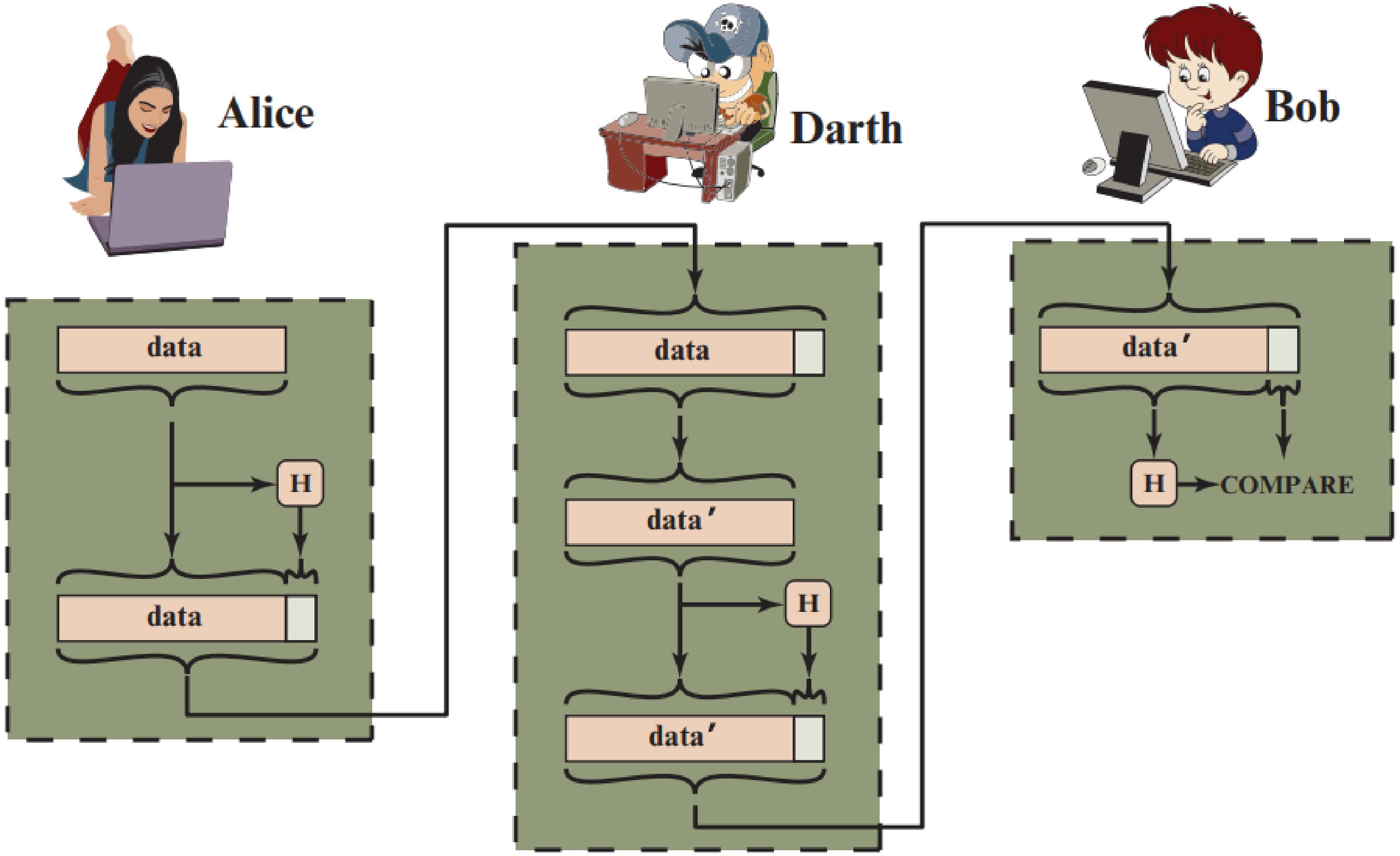
- Just listen to message
- No change!
- It’s a passive attack

Man-in-the-middle attack

- Change the original message
- It’s an active attack

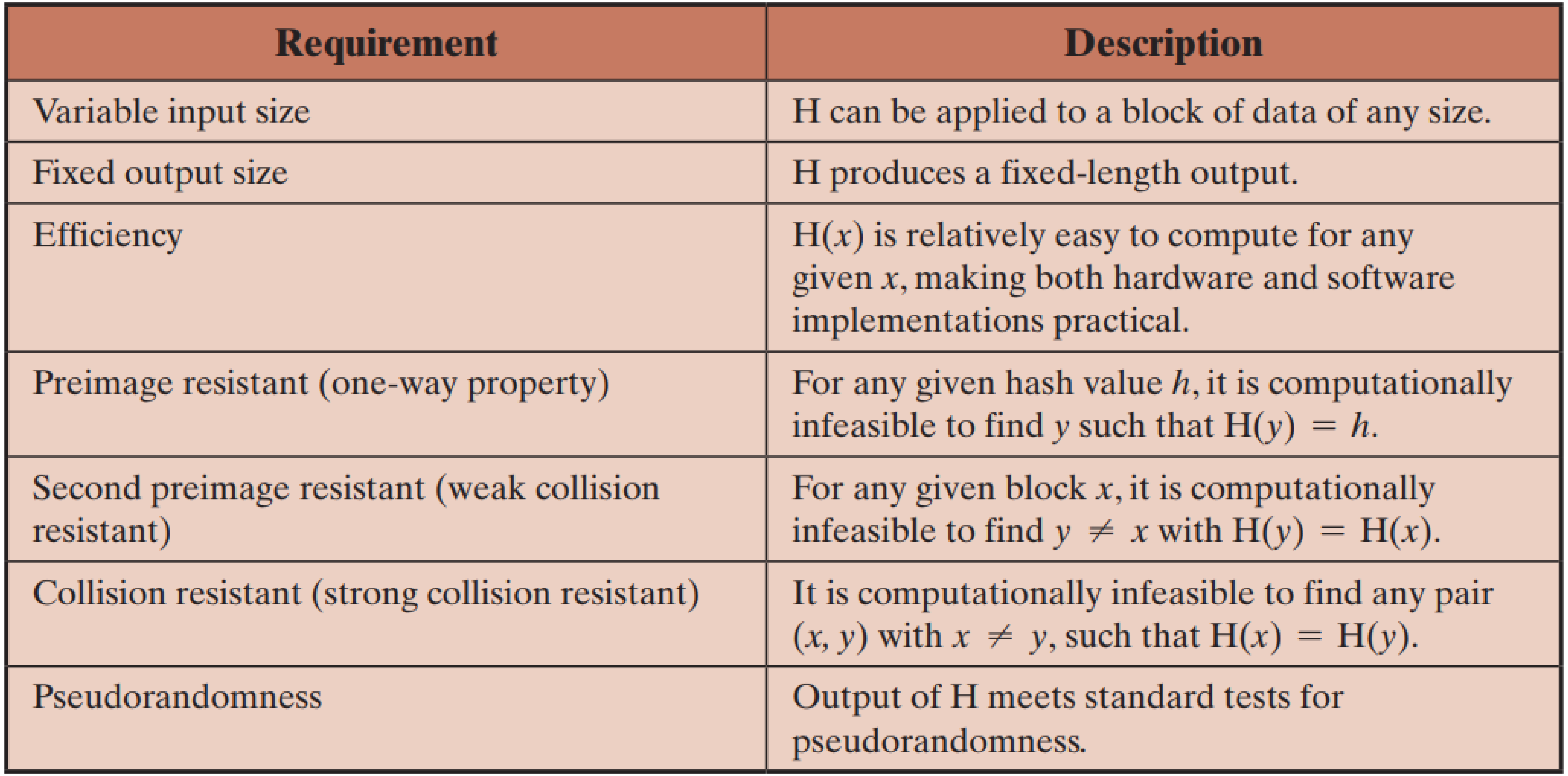

MITM Attack
CYBERVIE.COM
A CYBER SECURITY FIRM
Bob
Alice
Bob send msg to alice
Chuck send the msg to Alice
Chuck
Chuck intercepts and read Bob's message before it goes to alice

Attack Against Hash Function



(b) Man-in-the-middle attack

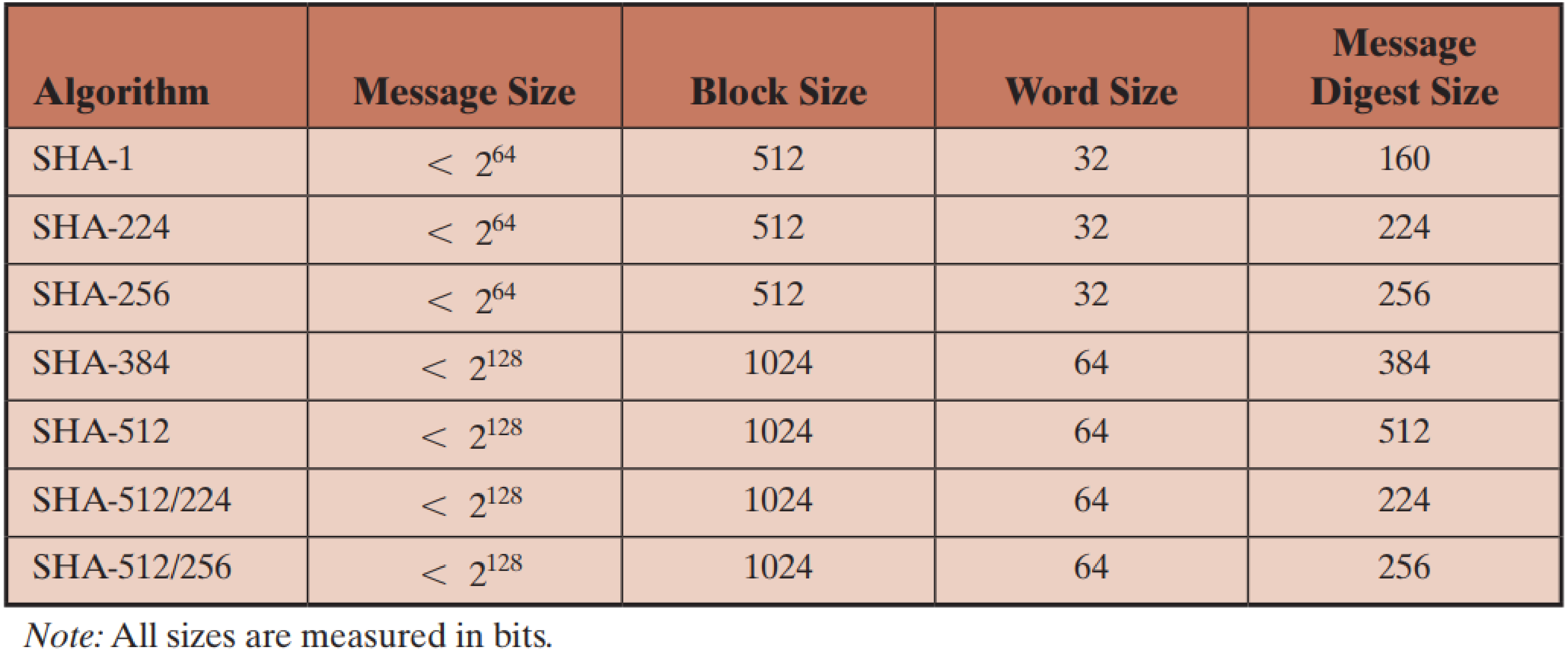
Requirements for a Cryptographic Hash Function H



Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) with $x \neq y$, such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness.

Secure Hash Algorithm (SHA)

- SHA was developed by NIST - 1993



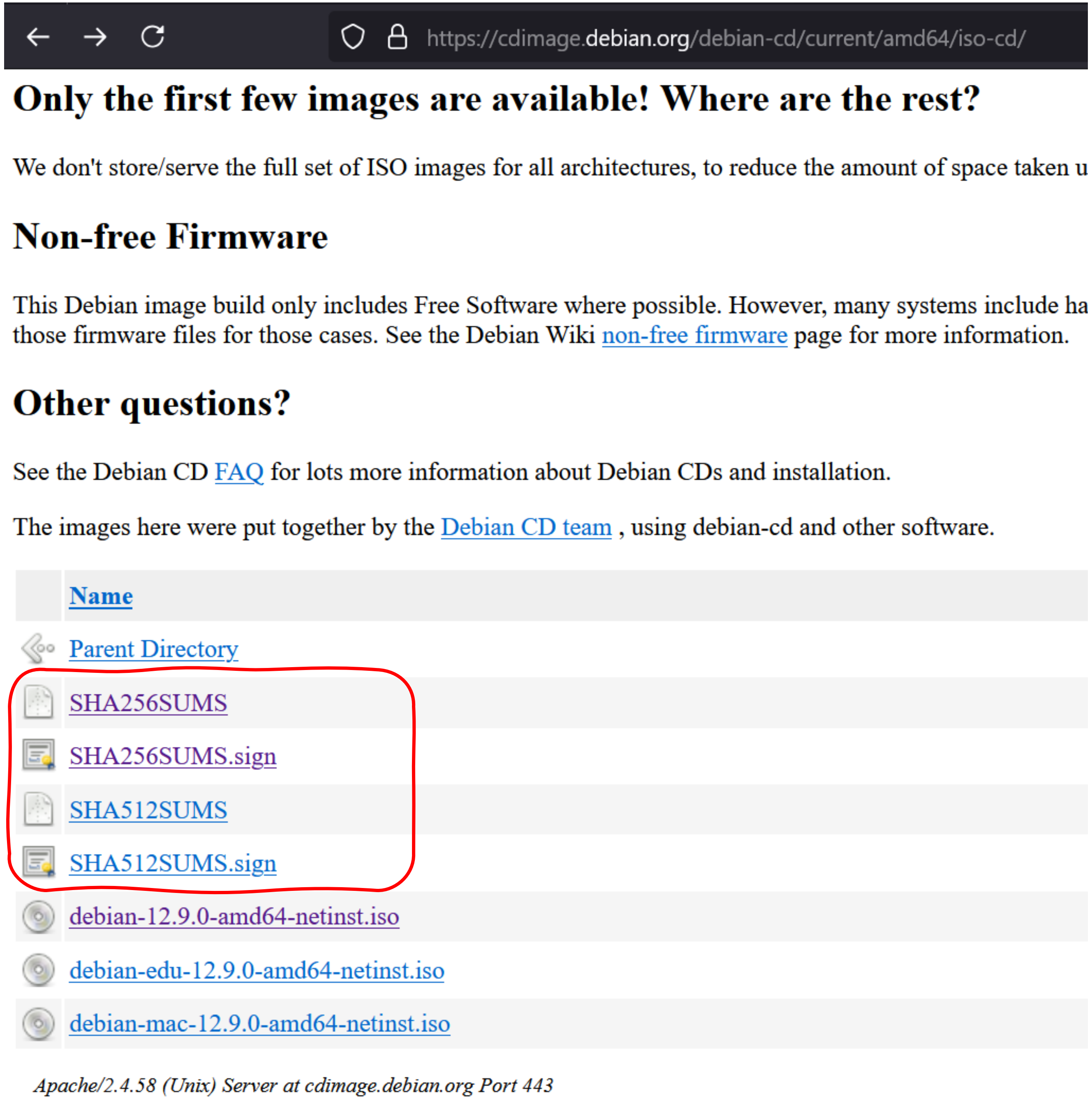
Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Note: All sizes are measured in bits.

Applications Of Hash Functions

- Message authentication is achieved using a message authentication code (MAC).
- File Integrity check.

File integrity check



https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/

Only the first few images are available! Where are the rest?

We don't store/serve the full set of ISO images for all architectures, to reduce the amount of space taken u

Non-free Firmware

This Debian image build only includes Free Software where possible. However, many systems include ha those firmware files for those cases. See the Debian Wiki non-free firmware page for more information.

Other questions?

See the Debian CD FAQ for lots more information about Debian CDs and installation.

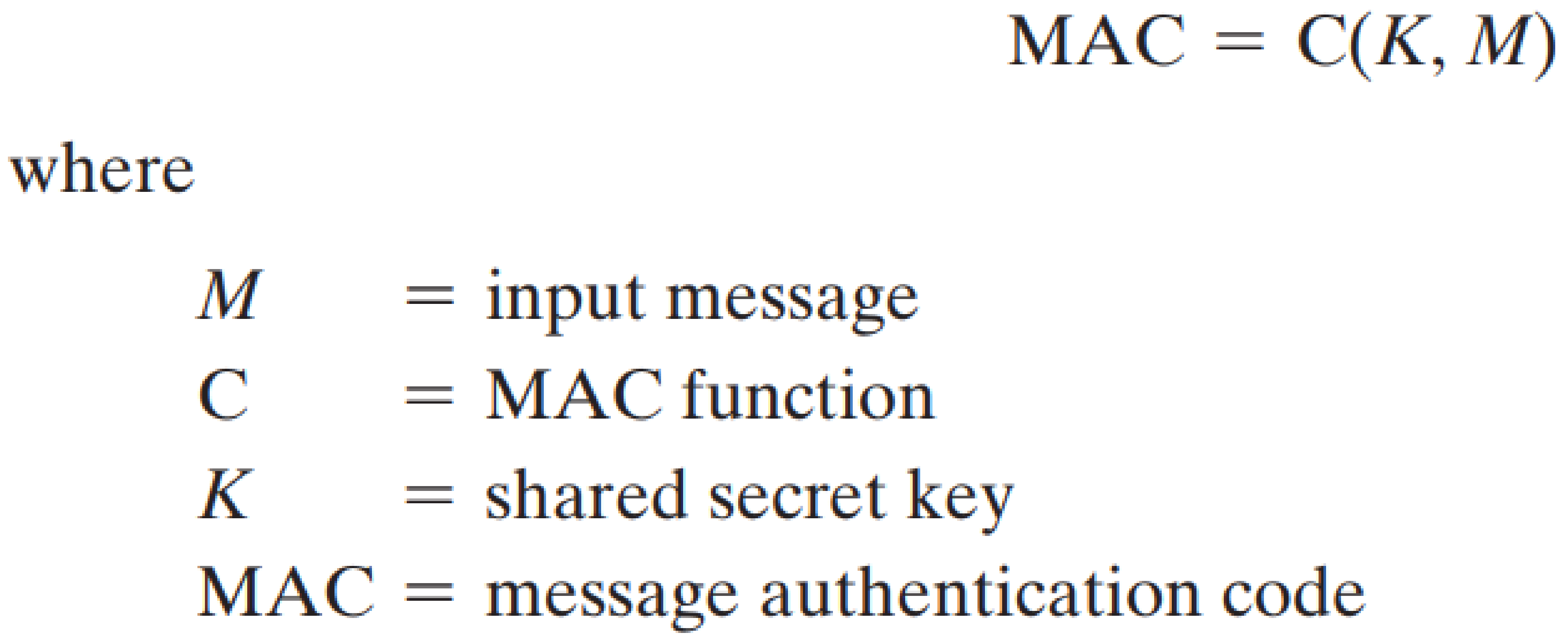
The images here were put together by the Debian CD team , using debian-cd and other software.

Name
Parent Directory
SHA256SUMS
SHA256SUMS.sign
SHA512SUMS
SHA512SUMS.sign
debian-12.9.0-amd64-netinst.iso
debian-edu-12.9.0-amd64-netinst.iso
debian-mac-12.9.0-amd64-netinst.iso

Apache/2.4.58 (Unix) Server at cdimage.debian.org Port 443

Message authentication code (MAC)

- A function of the message and a secret key that produces a fixed-length value that serves as the authenticator.



$$\text{MAC} = \text{C}(K, M)$$

where

M = input message

C = MAC function

K = shared secret key

MAC = message authentication code

Message Authentication Code - MAC

- The message plus MAC are transmitted to the intended recipient.
- The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC.
- The received MAC is compared to the calculated MAC.

MACs BASED ON HASH FUNCTIONS: HMAC

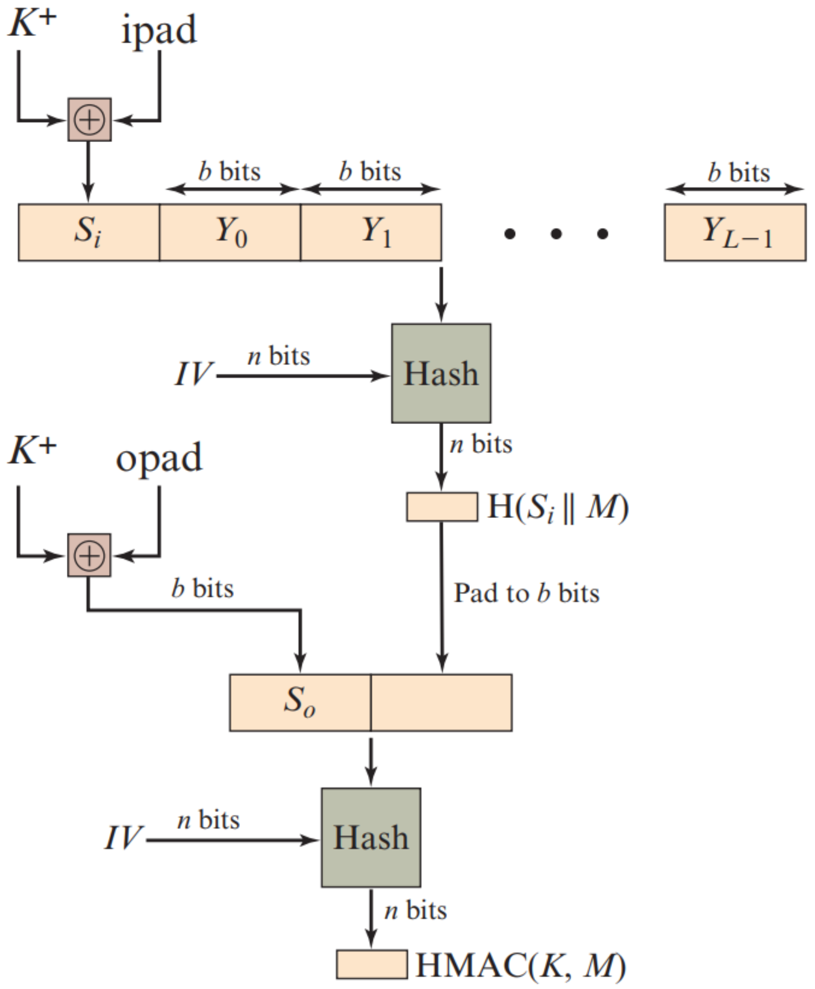


Figure 12.5 HMAC Structure